Machine Learning

Assignment 2 (Descriptive Questions)

Mohammad Rouintan
Student No: 400222042
Shahid Beheshti University

(Spring 1402)

Exercise 1:

Is it possible for an SVM classifier to provide a confidence score or probability when making predictions on a particular instance? Explain it.

Solution:

An ***SVM*** classifier can output the distance between the instance and the decision boundary, we can use this as a confidence score. However, SVM cannot provide probability but When the constructor option probability is set to True, class membership probability estimates(from the methods ***predict_proba()*** and ***predict_log_proba()***) are enabled. In the binary case, the probabilities are calibrated using Platt scaling: logistic regression on the SVM's scores, fit by an additional cross-validation on the training data. The same probability calibration procedure is available for all estimators via the ***CalibratedClassifierCV***.

Exercise 2:

What actions should you take if you have trained an SVMclassifier using an RBF kernel but notice that it underfits the training set? Would it be appropriate to increase or decrease the value of γ (gamma) or C,or both?

Solution:

Increasing gamma makes the function curve narrower, as a result of which the range of influence of each sample becomes smaller, that is, the decision boundary becomes more irregular and twists around the samples. On the other hand, the decreasing of gamma makes the function curve wider and the influence range of the samples larger. In this case, the decision boundary will be smoother, so gamma acts like the C hyperparameter. As a result, if the model underfits training set, we should ***increase*** γ or C or both.

Exercise 3:

What does it mean for a model to be ϵ-insensitive?

Solution:

In ***support vector regression***, we try to fit more samples inside the margin. The border width is controlled by the ϵ hyperparameter. Adding more samples inside the margin does not affect the prediction accuracy of this model, so it is called an ϵ-insensitive model. Then the idea is to use an ***insensitive tube*** in which the errors are ignored. ***ϵ-insensitive Tube*** means that any point in the dataset that falls within this tube will be ***disregarding*** the error. Basically we are allowing the margin of error for all the data points within the tube. So the residual or error is computed for the data points that lie outside the boundary of this ϵ-Insensitive Tube.

Exercise 4:

What is the difference between hard margin and soft margin SVM? When would you use each one?

Solution:

Hard Margin Classification:
If we apply strict restrictions according to which all samples must fall outside the margin, it is called hard margin classification. Hard margin classification has two major problems: one, it can only be used in cases where the data are linearly separable and two, it is sensitive to outliers.

Soft Margin Classification:
To avoid hard margin classification problems, it is better to use a more flexible model. The goal here is to find a good balance between the widest possible margin and limiting margin violations (i.e. samples that fall in the middle of the margin or even on the opposite side). This method is called soft margin classification.

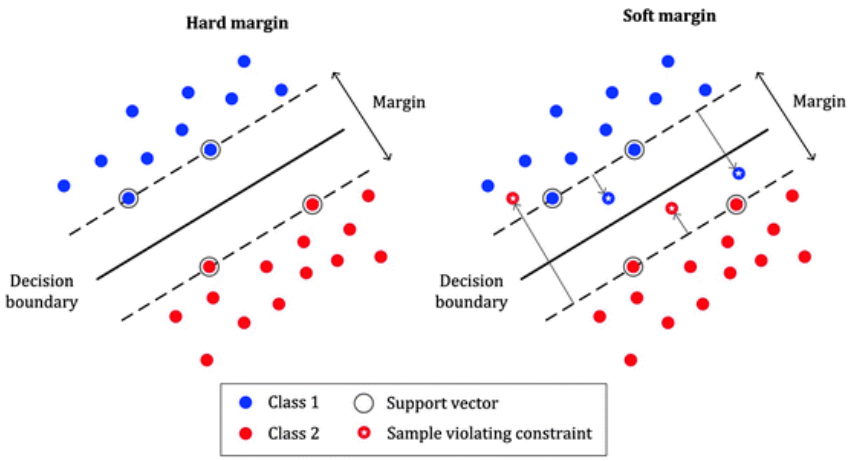

Figure 1: Hard Margin vs Soft Margin

The difference between a hard margin and a soft margin in SVMs lies in the separability of the data. If our data is linearly separable, we go for a hard margin. However, if this is not the case, it won't be feasible to do that. In the presence of the data points that make it impossible

to find a linear classifier, we would have to be more lenient and let some of the data points be misclassified. In this case, a soft margin SVM is appropriate.
Sometimes, the data is linearly separable, but the margin is so small that the model becomes prone to overfitting or being too sensitive to outliers. Also, in this case, we can opt for a larger margin by using soft margin SVM in order to help the model generalize better.

Exercise 5:

Is a node's Gini impurity generally lower or greater than its parent's? Is it generally lower/-greater,or always lower/greater?

Solution:

A node's Gini impurity is generally lower than its parent's. This is ensured by the ***CART*** training algorithm's cost function, which splits each node in a way that minimizes the weighted sum of its children's Gini impurities. However, if one child is smaller than the other, it is possible for it to have a higher Gini impurity than its parent, as long as this increase is more than compensated for by a decrease of the other child's impurity.

Exercise 6:

Is it a good idea to consider scaling the input features if a Decision Tree underfits the training set?

Solution:

One of the many advantages of the decision tree algorithm is that it does not require much data preparation. In fact, ***no rescaling or centralization*** is required to train this model. So if a Decision Tree underfits the training set, scaling the input features has no effect.

Exercise 7:

How can you use tree-based models for feature selection?

Solution:

Feature selection using ***Random Forest*** comes under the category of Embedded methods. Embedded methods combine the qualities of filter and wrapper methods. They are implemented by algorithms that have their own built-in feature selection methods. Some of the benefits of embedded methods are:

- They are highly accurate.
- They generalize better.
- They are interpretable.

To measure the importance of a feature, the Random Forest model looks at how much the nodes of the tree that use that feature reduce the average impurity (over all trees in the forest). More precisely, it is a weighted average where the weight of each node is equal to the number of training samples associated with it.
After training a forest, the model automatically calculates this score for each feature and then rescales it, so that the sum of all importance indices equals 1. We can use the ***feature_importance_*** to see this score

Exercise 8:

How do you tweak the hyperparameters of the following model in mentioned circumstances:

- AdaBoost-Underfitting
- Gradient Boosting-Overfitting

Solution:

AdaBoost-Underfitting:
If AdaBoost model underfits the training set, we can try increasing the number of estimators or reducing the regularization hyperparameters of the base estimator. We may also try slightly increasing the learning rate.

Gradient Boosting-Overfitting:
If Gradient Boosting model overfits the training set, we should try decreasing the learning rate. We could also use early stopping to find the right number of predictors.

Exercise 9:

What is the difference between homogeneous and heterogeneous ensembles? Which one is more powerful?

Solution:

Homogeneous:
Homogeneous ensemble consists of members having a single-type base learning algorithm. Popular methods like bagging and boosting generate diversity by sampling from or assigning weights to training examples but generally utilize a single type of base classifier to build the ensemble. In contrast to Heterogeneous Ensembles we will use a large amount of estimators for this algorithm. Note that the datasets for this model should be separately sampled in order to guarantee independence.

Heterogeneous:
Heterogeneous ensemble consists of members having different base learning algorithms such as SVM, ANN and Decision Trees. A popular heterogeneous ensemble method is stacking, which is similar to boosting. They usually work well if we have a small amount of estimators.

Homogeneous ensemble methods, use the same feature selection method with different training data and distributing the dataset over several nodes while Heterogeneous ensemble methods use different feature selection methods with the same training data.

Exercise 10:

How ROC and AUC are being used in the evaluation of classification performance?

Solution:

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the 'signal' from the 'noise.' In other words, it shows the performance of a classification model at all classification thresholds. The Area Under the Curve (AUC) is the measure of the ability of a binary classifier to distinguish between classes and is used as a summary of the ROC curve.

ROC:
The ROC curve displays the true positive rate (TPR, also known as the recall) in terms of the false positive rate (FPR). The higher the recall(TPR), the more false positives the model produces (i.e., the higher FPR). In a ROC curve, a higher X-axis value indicates a higher number of False positives than True negatives. While a higher Y-axis value indicates a higher number of True positives than False negatives. So, the choice of the threshold depends on the ability to balance False positives and False negatives. A good classifier tries to take the maximum distance (toward the upper-left corner) from the line representing the random classifier.

AUC:
One way to compare two classifier models is to measure their area under the curve or AUC. In a complete classifier, this area is equal to 1, while its value in a random classifier is 0.5.

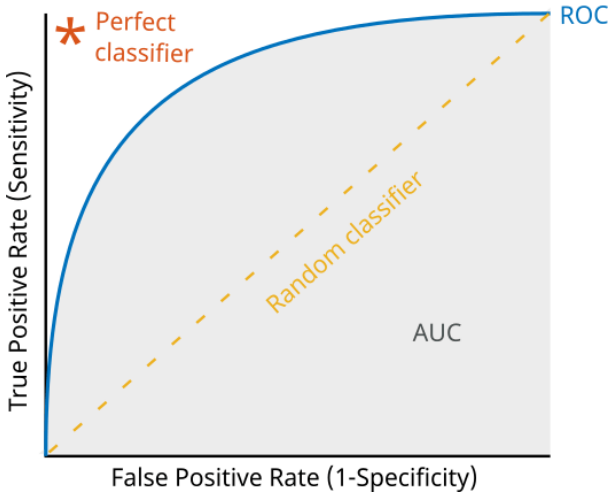

Figure 2: ROC curve and AUC

Exercise 11:

How does the threshold value used in classification affect the model's performance? This value specifies a cut-off for an observation to be classified as either 0 or 1. Can you explain the trade-off between false positive and false negative rates, and how the choice of threshold value impacts precision and recall?

Solution:

The classification threshold in ML, also called the decision threshold, allows us to map the sigmoid output of a binary classification to a binary category.
It can be risky to assume that the default 0.5 classification threshold is correct for a specific use case without proper model evaluation and analysis. The number and nature of model misclassifications will determine a machine learning initiative's success. Setting the appropriate classification threshold is essential to limiting these misclassifications, and therefore indispensable in ML.
Some signs to look out for that indicate that 0.5 may not be the best threshold value include:

- Predicted probabilities are not calibrated, i.e., they don't match the expected distribution of class probabilities.
- Training metrics and evaluation metrics differ.
- The distribution of classes is skewed, i.e., the positive class only appears for a small subset of data samples.
- The cost of an error varies significantly by misclassification type.

Various techniques exist to compensate for these cases by looking at improving the underlying data via preprocessing and augmentation, or by improving the algorithm itself with, for example, a custom loss function. Classification thresholding is often preferred in practice as more interpretable and seamless.
The classification threshold selection directly affects these values of the confusion matrix.

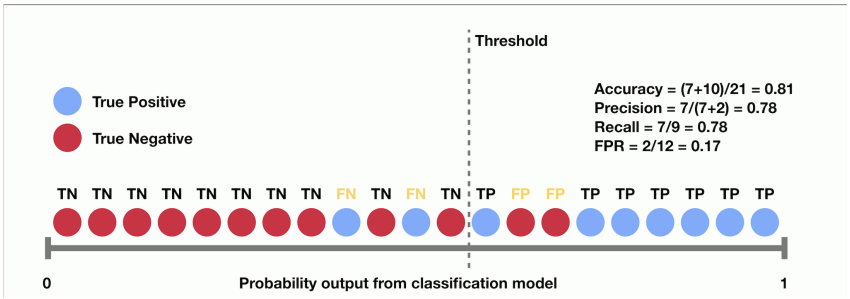

Figure 3: Example of Classification Threshold

All ML classification thresholds are specific to their use case and must be tuned for each ML application individually. To choose the best classification threshold, we need to define what types of mistakes are least problematic to our use case. While no ML model is perfect, we can make smart choices to optimize performance. One such optimization choice is tuning the classification threshold.

Exercise 12:

What is the difference between one-vs-one and one-vs-all multiclass classification approaches in classifiers? Under what circumstances would you use one over the other?

Solution:

One-Vs-One:
Suppose we want to classify the data into k classes. We can build $\binom{k}{2}$ SVM for this task, where each SVM is to separate the data between two classes. For example, one of the SVMs can be used to compare class i (with code +1) and class j (with code -1). To make a decision about a data, after applying all SVMs on it, we count the number of times each class is selected and finally assign the data to a class that is selected more times.

One-Vs-All:
In this process, we build up to k SVMs, each SVM separates one of the classes against $k-1$ other classes. Let $\beta_{0i}, \beta_{1i}, \ldots, \beta_{pi}$ be the parameters of the SVM that separates the data of the i-th class (with code +1) from the data of other classes (with code -1). If x^* is a new test data, we assign this data to the class with the highest $\beta_{0i} + \beta_{1i}x_1^* + \cdots + \beta_{pi}x_p^*$ value.

One-Vs-One has one major advantage: each classifier only needs to be trained on a small part of the original dataset (the two classes it needs to distinguish from each other), not the entire dataset. Some algorithms (such as support vector machine) do not have good scalability, that is, their efficiency drops drastically as the dataset grows. For these types of algorithms, a One-Vs-One approach is preferable, because training a large number of classifiers with small datasets will be much faster than training one classifier with a large dataset. However, for most binary classification algorithms, One-Vs-All method is preferable.

Exercise 14: (Extra Point)

How can you use SVM for anomaly detection? What are the challenges in using SVMfor anomaly detection? (Extra Point)

Solution:

We have a dataset and know that a few points are the outliers and we want to detect them. The ***One-Class SVM*** can be helpful in this circumstance so suppose we have a set of points and a few outliers our goal is finding a separator between the outliers and the regular set of points.
A One-Class Classification method is used to detect the outliers and anomalies in a dataset. Based on Support Vector Machines (SVM) evaluation, the One-Class SVM is an unsupervised model for anomaly or outlier detection. Unlike the regular supervised SVM, the One-Class SVM does not have target labels for the model training process. Instead, it learns the boundary for the normal data points and identifies the data outside the border to be anomalies.

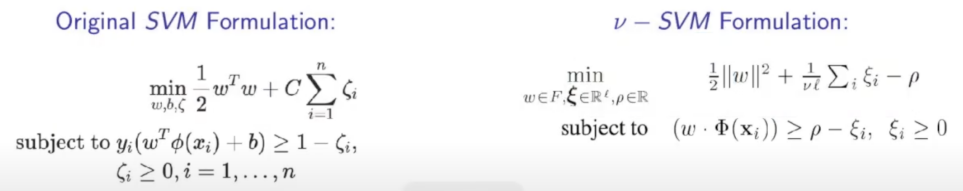

Figure 4: Formulation

Important HyperParameter of One-Class-SVM:

- $\nu(nu)$ is to specify the percentage of anomalies. For example $nu = 0.02$ means that we have around 2% outliers in the dataset.
- $\gamma(gamma)$ is a kernel coefficient, and it is for 'rbf', 'poly', and 'sigmoid' kernels.
- Kernel specifies the kernel type. The radial basis function (rbf) kernel is a commonly used kernel type. It maps data from a low dimensional space to a high dimensional space to help the SVM model draw a decision boundary.

Exercise 16: (Extra Point)

How do you handle the class imbalance in Ensemble Learning? Provide some techniques and explain their working. (Extra Point)

Solution:

Imbalanced Data affect the learning performance of algorithms in Machine Learning. The boundary of decision in out of balance data chosen by most standard algorithms of machine learning tends to bias toward the ***majority*** class and hence misclassifies the ***minority*** class.

The main objective of ensemble methodology is to improve the performance of single classifiers. The approach involves constructing several two stage classifiers from the original data and then aggregate their predictions.

Bagging Techniques:
In this method, we use a single training algorithm, but we train each predictor on different random subsets of the training dataset. When random sampling with replacement is done, it is called ***Bagging***.

Boosting Techniques:
Any combination method that can combine several weak learners and make a strong learner is called a ***Boosting***. The main idea of most boosting methods is to train several predictors sequentially, so that each predictor corrects the model before it. There are many boosting methods, some of which we have mentioned below.

1. ***AdaBoost Technique:***

   Ada Boost is the first original boosting technique which creates a highly accurate prediction rule by combining many weak and inaccurate rules. After each round, it gives more focus to examples that are harder to classify. The quantity of focus is measured by a weight, which initially is equal for all instances. After each iteration, the weights of misclassified instances are increased and the weights of correctly classified instances are decreased.

2. ***Gradient Boosting Technique:***

   In Gradient Boosting many models are trained sequentially. It is a numerical optimization algorithm where each model minimizes the loss function, using the Gradient Descent Method. Decision Trees are used as weak learners in Gradient Boosting. Gradient Boosting builds the first learner on the training dataset to predict the samples, calculates the loss (Difference between real value and output of the first learner). And use this loss to build an improved learner in the second stage.

3. ***XG Boost Technique:***

   XGBoost (Extreme Gradient Boosting) is an advanced and more efficient implementation of Gradient Boosting Algorithm. It is 10 times faster than the normal Gradient Boosting as it implements parallel processing. It is highly flexible as users can define custom optimization objectives and evaluation criteria, has an inbuilt mechanism to handle missing values. Unlike gradient boosting which stops splitting a node as soon as it encounters a negative loss, XG Boost splits up to the maximum depth specified and prunes the tree backward and removes splits beyond which there is an only negative loss.